ATTORNEY'S DOCKET:
066101.0316

PATENT APPLICATION

A SYSTEM AND METHOD FOR DYNAMICALLY CONSTRUCTING PACKET CLASSIFICATION RULES

TECHNICAL FIELD OF THE INVENTION

This invention relates generally to computer networks, and more specifically relates to a system and method for dynamically constructing rules to classify packets transmitted across a network.

BACKGROUND OF THE INVENTION

Packet based computer networks transmit information in packets that are formatted with a sequence of well-known header fields that direct the packets through the network. For example, a TCP/IP packet on an Ethernet network consists of three parts, an Ethernet header, an IP header, and a TCP header. The Ethernet header in turn includes three well-known fields. The source address, the destination address and the "EtherType" field. From the EtherType field the format of subsequent data may be determined. For instance, if the EtherType field indicates the packet contains an IP datagram, packet field values for the IP datagram allow the determination of source address, destination address and protocol fields. The protocol field identifies the type of data that follows, such as TCP, UDP, etc. The packet header information is used by network computer devices to route data through the network.

Network computing devices perform routing and switching functions based upon computations performed on the packet field values. For instance, router software on a general purpose computer sends packets to different output network interfaces based upon computations performed from header packet field values. To improve network transmission speeds, special purpose devices are used to perform simple, well specified functions at high speeds that direct network traffic. For instance, network appliances such as routers, switches and firewalls perform fixed functions based on one or more fixed fields using hard wired instructions that process packets in a substantially more rapid manner than

10034435.122701

software functions. As an example, a router computes the output interface for a packet based on the packet's destination address in the IP header.

Although hard wired instructions, such as those defined in application specific integrated circuits (ASICs) provide for rapid processing of packets through a network, ASIC designs are typically inflexible since the hard wired instructions generally cannot be reprogrammed through software. Thus, for example, networks that rely on routers have difficulty implementing services which generally call for varying packet processing behaviors since router functions are generally hard wired into ASICs. For instance, internet service providers that provide customer access to the Internet over router based networks have difficulty deploying services that provide for individual handling of packets related to specific customers.

In order to aid in the deployment of services to packet based networks, programmable network processors have been developed for use in network appliances such as routers, switches and firewalls. Network processors run a software program that determines the processing of packets but handles packets in a rapid manner by performing certain functions specific to processing of network packets through hardware implementations. For example, network processors support table look-up operations with hard wired instructions allowing routing functions that rely on table look-ups to occur at rates much faster than available through general purpose processors. Network processors support programs that look at packet field values and perform table look-up

10034435.122701

operations to determine the processing for the packet. For example, a program on a network processor classifies a packet by using information from IP source and destination address field values. The fields examined and the combination of the fields are determined by the program loaded on the network processor.

One difficulty with network processors is that loading a program on a network processor takes several seconds and brings the network processor off line so that packets are either dropped or passed through the network processor without processing. Thus, as an example, a network processor used in a router will not route packets while a new program is loaded. In systems that use fixed combinations of fields to process packets, the programming limitation of network processors does not present a substantial difficulty since the program running on the network processor need not change very often. However, in order to provide services to packet based networks, such as with the programmable network nodes disclosed in U. S. Patent Application Serial Number ________________ (Attorney Docket No. 066101.___) entitled "System and Method for Programming Network Nodes," which is incorporated herein by reference, the program running on the network processor may have to change more often.

Another difficulty with programming network processors is that newly added network processor programs must continue to process packets at line rates to avoid degrading network operations. For instance, if a program on a network processor fails to process packets at line rates, packets will be dropped and network performance severely degraded. The addition of new classifiers to a

10034435.122701

network processor program has an unpredictable effect on the speed at which the program operates on the network processor. Thus, especially in the case of complex packet processing behavior implementations, the reliability of new network processor programs is difficult to predict.

10023435.122701

SUMMARY OF THE INVENTION

Therefore, a need has arisen for a system and method which dynamically constructs rules in a network processor while the network processor remains on line.

A further need has arisen for a system and method which provides flexibility for network processors to enable a variety of packet processing behaviors in a reliable manner at line speed.

A further need has arisen for a system and method which allows construction of packet classifiers in a dynamic manner that supports deployment of services to a packet based network.

In accordance with the present invention, a system and method is provided that substantially eliminates or reduces disadvantages and problems associated with previously developed systems and methods for classifying packets transferred across a packet based network. A program selects predetermined packet field values and classifies packets by matching one or more packet field values with a data structure. New packet classifications are created by updating the data structure to associate the one or more predetermined packet field's values with the new packet classification without changing the program.

More specifically, a network processor runs a programmably fixed program that supports dynamic creation of packet classifiers through exploitation of high speed network processor table look-up operations. The dynamic classifiers are arbitrary Boolean combinations of values from packet fields extracted from network packet headers by the program of the network processor. Packet

10043435.122701

processing behaviors are added as new rules by modifying tables within a data structure while maintaining the underlying network processing code unchanged.

In one embodiment, the network processor program parses packets in a predetermined order encoded as a parse tree on the network processor. Each node of the parse tree identifies packet fields examined and each branch of the parse tree indicates the value of extracted fields. The network processor examines packet field values according to the parse tree programming to extract and save useful information. When a leaf node of the parse tree is reached, a transmit function is called that uses one or more of the captured field values to compute a classification destination identification (DID) for the packets.

The packet classification computed by the transmit function results from the matching of relevant field values with a data structure to compute the destination identification. The transmit functions use network processor table look-up functions to perform matching between pattern trees and ordered virtual trees. The pattern tree match identifies the longest match value and provides a virtual handle for use in the ordered virtual tree. The transmit function then matches the virtual handles against the ordered virtual tree data structure to compute the destination identification for classification of the packet. The pattern trees match values extracted from pattern fields and the ordered virtual trees match combinations identified with virtual handles from the pattern tree. In this manner, arbitrary Boolean combinations of extracted header fields can be

10023435.122701

formed to provide high speed and fine grained classification rules which are dynamically added or deleted through modification of table values without changing the parse tree program and causing disruption of service.

In the operation of one embodiment, the network processor parsing program and data structure run on a pattern processor to classify packets transmitted through a network. For instance, a programmable node includes a pattern processor and system interface that allows prioritized rules to classify packet flows, such as to enable a service. The rules are sets of properties of packets identified by an associated identifier. The packet values for enabling the rules are extracted from packet fields by the parsing program. Rules are installed without a need for modifying the parsing program by instead updating the data structure pattern trees and ordered virtual trees. The dynamic insertion and deletion of values from pattern trees and ordered virtual trees are performed without interrupting network processor operations.

The present invention provides a number of important technical advantages. One important technical advantage is that rules are dynamically constructed in a network processor for classification of packets while the network processor is on-line. The dynamic construction of rules by updating and modifying the data structure while leaving the network processor program unchanged avoids service interruption while allowing classification rules to involve arbitrary Boolean combinations of header fields. The programmably fixed parsing program of the

10023455.102701

network processor allows the creation of rules at a rate of several thousand rules per second on current generation network processor hardware since the fixed program extracts desired field values from packets in a consistent manner and enables changes to packet processing behaviors by modifying data structures instead of network processor programming.

Another important technical advantage of the present invention is that packet processing behavior rules are rapidly constructed and used on a programmable node to classify packets by arbitrary Boolean combinations of header fields without risk of slowing down network processor operations below line speed. In addition to bring a network processor off line, reprogramming of a network processor to handle new classifiers may have a varied impact on network processor performance. Thus, by maintaining a programmably fixed network processor program that handles new classifications with updates to data structures instead of changing the program, the present invention provides for dynamic construction of packets processing behaviors without risk of slowing down network processor operations. The "fixed" network processor program may be carefully tested to ensure that it maintains line rates with any combination of rules.

Another important technical advantage of the present invention is that programmable nodes may update services dynamically and on a real time basis with minimal impact on the performance of a packet based network. Such services may rely on classification rules involving arbitrary Boolean combinations of header fields, such as directing identified packet flows from a source to a

10034435.122701

destination at a predetermined service level or with other desired packet processing behaviors, including blocking undesired flows like pornography or Napster, and forwarding identified flows to predetermined queues or paths.

10034254.122701

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the present invention and advantages thereof may be acquired by referring to the following description taken in conjunction with the accompanying drawings, in which like reference numbers indicate like features, and wherein:

FIGURE 1 depicts a block diagram of a programmable node that provides services to a packet based network;

FIGURE 2 depicts a block diagram of a pattern processor for classifying packets with a programmably fixed parse tree program; and

FIGURE 3 depicts one embodiment of a parse tree for extracting packet field values from an Ethernet packet.

10/27.55.1ƐƐ00T

DETAILED DESCRIPTION OF THE INVENTION

Preferred embodiments of the present invention are illustrated in the FIGURES, like numerals being used to refer to like and corresponding parts of the various drawings.

Services are difficult to deploy in packet based networks since the service specific handling of packets, requires the inspection of packet header fields and, in some cases, data. For instance, some types of service specific handling are specific routing, firewall functions based on content, bandwidth shaping to delay, prioritize or drop packets, and MPLS LERs to add tags to packets. One alternative for performing services on a packet based network is to hard wire instructions for the service at nodes in the packet based network. However, hard wired instructions are difficult to deploy and change. Software based services generally do not operate at fast enough line speeds unless enhanced by specialized hardware. Network processors offer the advantage of combined software and hardware designs that specialize in processing packets. However, changing the program on a network processor to enable a new packet processing behavior generally requires taking the network processor off line for several seconds, thus making reprogramming of the network processor an impractical alternative when seeking to deploy services on a rapid and real time basis.

Referring now to FIGURE 1, a block diagram depicts a programmable node that uses network processors to dynamically create rules for providing services to packet based networks. A programmable node 10 includes one or

10034435 . 122701

more network processors that classify, modify, shape and route packets through the packet based network. Network processors are functionally divided into three parts, a pattern processing (PP) function 12, a routing\switch processing (RSP) function 14 and a system interface (SI) function 16. These three functions may be handled by a single network processor chip or separate chips as depicted by Figure 1.

Packets received from a physical interface 18 are passed to pattern processor 12. Physical interface 18 may include interfaces for Ethernet, Sonet, ATM, RPR (802.17), TDM (T1, T3, DS3, E3) and other types of physical networks that transmit Internet traffic. Pattern processor 12 classifies the packet according to a programmed set of rules and sends the packets along with a classification destination identifier (DID) to routing\switch processor 14. The routing\switch processor 14 uses the destination identifier, which indicates the result of the classification step, to modify, shape and route the packet to an appropriate output physical interface 18. System interface 16 installs new rules and programs in the pattern processor 12 and routing\switch processor 14.

The dynamic construction of rules for packet processing behaviors are provided with a network processor program 20 and data structure 22. A host processor 24 programs the network processor program 20 and data structure 22 through system interface 16 to perform rules that compute destination identifications associated with a packet based on information in the packet header and then modify, shape or route the packets

10043724.103101

according to the destination identification through routing\switch processor 14. Network processor program 20 is programmably fixed so that rules are dynamically constructed with table modifications without having to change network processor program 20, thus avoiding bringing the programmable node 10 off line. Instead, network processor program 20 extracts predetermined packet field values in a programmed but fixed manner with classifiers dynamically created by updating data structure 22 instead of network processing program 20. The fixed nature of network processor program 20 exploits the table look-up operations available in network processors to maintain high speeds and uses dynamic classifiers of arbitrary Boolean combinations of well known header field values of the network packets by updating data structure 22. Some examples of header fields extracted include: MPLS label, time to live, EXP bits and BS; Ethernet source, destination MAC address, EtherType, 802.1p priority, 802.1q VLAN identifier and 802.1q CFI; UDP source/dest port and length; ICMP type, code, type-specific data; IP type of service, dont fragment flag, protocol, time to live; and TCP flags (SYN, FIN, ACK, URG, PSH, RSI) and length window size.

Host processor 24 prioritizes service rules so that programmable node 10 classifies packet flows for processing though the network. The prioritized rules used by host processor 24 are a set of properties of the packet that identify packets and associate packets with processing behaviors. As an example, a processor rule might identify all packets with a predetermined TCP destination port and destination IP address so that those

10034435.122701

identified packets may be processed through the network in a desired manner, such as with a predetermined service level, bandwidth allocation or path. Alternatively, a service may block transfer of such identified packets, such as with pornography protection or undesirable programs such as Napster. The processing rules provided by host processor 24 have an associated identifier, such as a 20-bit or even 64-bit identifier that allows a large number of packet processing behaviors to be programmed.

Referring now to FIGURE 2, a block diagram depicts one embodiment of a static network processor program that allows new packet behavior rules to be installed on a programmable node 10. A network processor parse tree program 26 is a static program that remains unchanged even while new packet behavior rules are installed, with the new rules implemented through modifications made to a pattern tree data structure 28 and an ordered virtual tree data structure 30. Pattern tree data structure 28 and ordered virtual tree data structure 30 each allow for dynamic insertion and deletion operations without interrupting pattern processor 12's operation. A data structure modifier 32 performs dynamic insertion and deletion operations on pattern tree data structure 28 and ordered virtual tree data structure 30 to implement rules dynamically by modifying data. Network processor program modifier 34 allows changes to network processor parse tree program 26, although such changes generally require that pattern processor 12 be brought off line. Thus, network processor parse tree program 26 is programmably fixed so that during dynamic creation of rules the programmable node 10 remains on line by keeping the

10027155.122701

program fixed, but the program may be modified if needed by bringing the pattern processor off line.

Pattern trees are tables with one or more entries that represent patterns and contain a bit mask. Network processors, such as those available from Agere, have special hardware for using pattern trees. When a pattern tree is searched for a value, the longest match found is used with the bit mask indicating which bits of the pattern are significant. For example, a search for the value 192.208.12.14 in the following pattern tree would return three matches, rows 1, 3 and 4, with row 3 having the longest match of 32 bits.

Pattern	Mask	ID
192.208.12.0	FFF0	1
10.0.0.0	F000	2
192.208.12.14	FFFF	3
0.0.0.0.	0000	4

Data structure modifier 32 updates the pattern tree with patterns, masks and identifiers known as virtual handles. For instance, to dynamically create a service rule associated with an IP address, data structure modifier 32 inserts the IP address as a pattern in pattern tree data structure 28. Although the present embodiment uses a pattern tree data structure, alternative embodiments use other data structures that implement longest prefix matches.

Ordered virtual tree data structure 30 supports dynamically constructed rules by allowing combinations of multiple patterns in a single search. Network

10/22/01 10034435

processors, such as those available from Agere, have special hardware for using ordered virtual trees, also known as OV trees. Ordered virtual trees define the order in which rules are matched and are typically used to implement access control lists such as those found in a firewall. The ordered virtual tree is searched in order from top downward until a first match is made at which time the search is complete. In contrast, pattern trees find a longest prefix match with a search of the entire pattern tree to determine the most specific match, whereas ordered virtual trees need search only for the first match. Although the present embodiment uses an ordered virtual tree data structure, alternative embodiments use other data structures that implement first matches.

Referring now to FIGURE 3, an example of a parse tree is depicted that shows a parsing order for Ethernet packets. The parsing order is programmable into network processor parse tree program 26. The parse tree depicts how the pattern processor 12 running the network processor parse tree program 26 will examine Ethernet packet fields, with each node in the parse tree listing packet fields examined and each branch indicating the value of a particular field. The parse tree parses packets from the most general header information to more specific header information and extracts useful pieces of header field values, such as source address or VLANID and stores these values for use. When a leaf node of the parse tree is reached, a transmit function is called that uses the captured header field values to compute a destination identification for the packet.

10723755.122701

For example, the parse tree depicted by FIGURE 3 parses the indicated Ethernet packet header information to extract the ether type, follow the IP branch, extract the source address and destination address, follow the TCP branch, extract the source port and destination port and then call the "Do TCP" transmit function. The TCP transmit function uses the captured field values to compute a destination identification for the packet by matching the relevant field values with the data structure. For instance, the captured field values are matched against pattern trees to extract corresponding virtual handles based on the longest prefix match and then the virtual handles are matched against the ordered virtual tree to compute the destination identification. Thus, as depicted in FIGURE 2, the network processor parse tree program 26 matches parsed values against pattern tree data structure 28 to obtain virtual handles and then matches the virtual handles against ordered virtual tree data structure 30 to obtain destination identification values. The destination values are provided along with the packet to the routing\switch processor 14, which handles the packets according to a processing behavior associated with the destination identification programmed from host processor 24 to system interface 16 into routing\switch processor 14.

Advantageously, new rules for packet processing behaviors, such as rules that enable a service, are added by modifying the pattern tree data structure 28 and ordered virtual tree structure 30 while the underlying network processor parse tree program 26 remains unchanged. For example, with the destination address

10023435.122701

illustrated by FIGURE 3, in order to add a rule to "match all packets sent to 192.208.12.*, port 80" an entry is created in a TCP_DESTADDR pattern tree for the destination address of 192.208.12.* and another entry is created in the TCP_DESTPORT pattern tree for the destination port 80. The combination of the destination address and destination port are found with the TCP OV Tree for the combination of the destination address and the destination port. The pattern trees and ordered virtual trees are encoded dynamically with conventional techniques, such as those described in "A Tree-Based Packet Routing Table for Berkley Unix," by K. Sklower, Proceedings of the 1991 Winter USENIX Technical Conference, January, 1991. Network processors available from various manufacturers provide various levels of support for functions such as table look ups. The pattern trees and ordered virtual trees are initialized with a single entry matching so that if no other matches occur, the initialized entries identify a default destination identification associated with unmatched packets.

Although the present invention has been described in detail, it should be understood that various changes, substitutions and alterations can be made hereto without departing from the spirit and scope of the invention as defined by the appending claims.

10423545.121801